In-App Purchase on iPhone

Session 122

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

In-App Purchase on iPhone

Payam Mirrashidi
Director of Engineering, iTunes Store

Agenda

- Overview of In App Purchase
- Implementing In App Purchase
- Financial reports

Overview

What Is In App Purchase?

- Payment collection engine for iPhone apps
  - Requires iPhone OS 3.0
  - For paid apps only
- Facilitates In App Purchase
  - Built-in app features
  - Remote content and services
- App Store collects the payment
  - Same purchase experience and receipt
  - Updated iTunes Connect reports

What Are You Going to Sell?

- Content
- Functionality
- Services
- Subscriptions

What Are the Rules?

- Digital items only—no physical goods or services
- Delivered to app from which it was purchased
- Available on all devices

Supported Content Examples

- ✓ Digital books
- ✗ Physical books
- ✓ Virtual poker chips current game
- ✗ Virtual poker chips for other games
- ✓ In-game tool or accessory
- ✗ In-game credits for virtual goods

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis sodales, mauris ac dictum pellentesque, ligula arcu posuere leo, id placerat nibh sem non risus. Sed faucibus, leo eu vehicula tincidunt, eros augue

Supported Functionality Examples

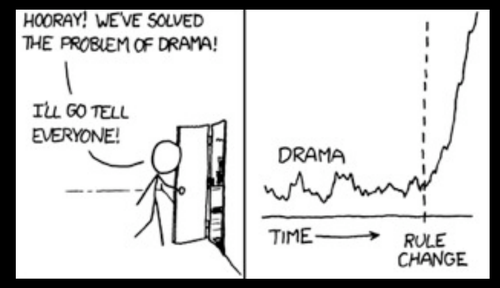
- Access to more game levels
- Ability to save
- Ability to manage additional itineraries

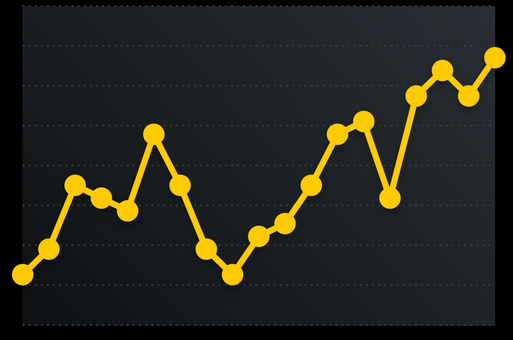
Supported Service Examples

- ✓ Business card scan service
- ✓ Voice transcription
- ✓ VoIP telephone service
- ✗ Plumbing repairs
- ✗ Courier delivery
- ✗ Web design services

Supported Subscription Examples

- ✓ Monthly periodical
- ✓ Daily comic strip
- ✓ Financial data and information
- ✗ Local video store membership
- ✗ Gym membership renewal





How Are Items Delivered?

- Built-in items
  - Additional features initially disabled within the app
  - Unlocked after purchase
- Server delivered items
  - Items not included in the original app
  - Remotely implemented or hosted and delivered by a server
- Combination of both

Implementation

Implementing In App Purchase

- Define metadata and upload to iTunes Connect
- Build purchase workflow
- Test in sandbox environment

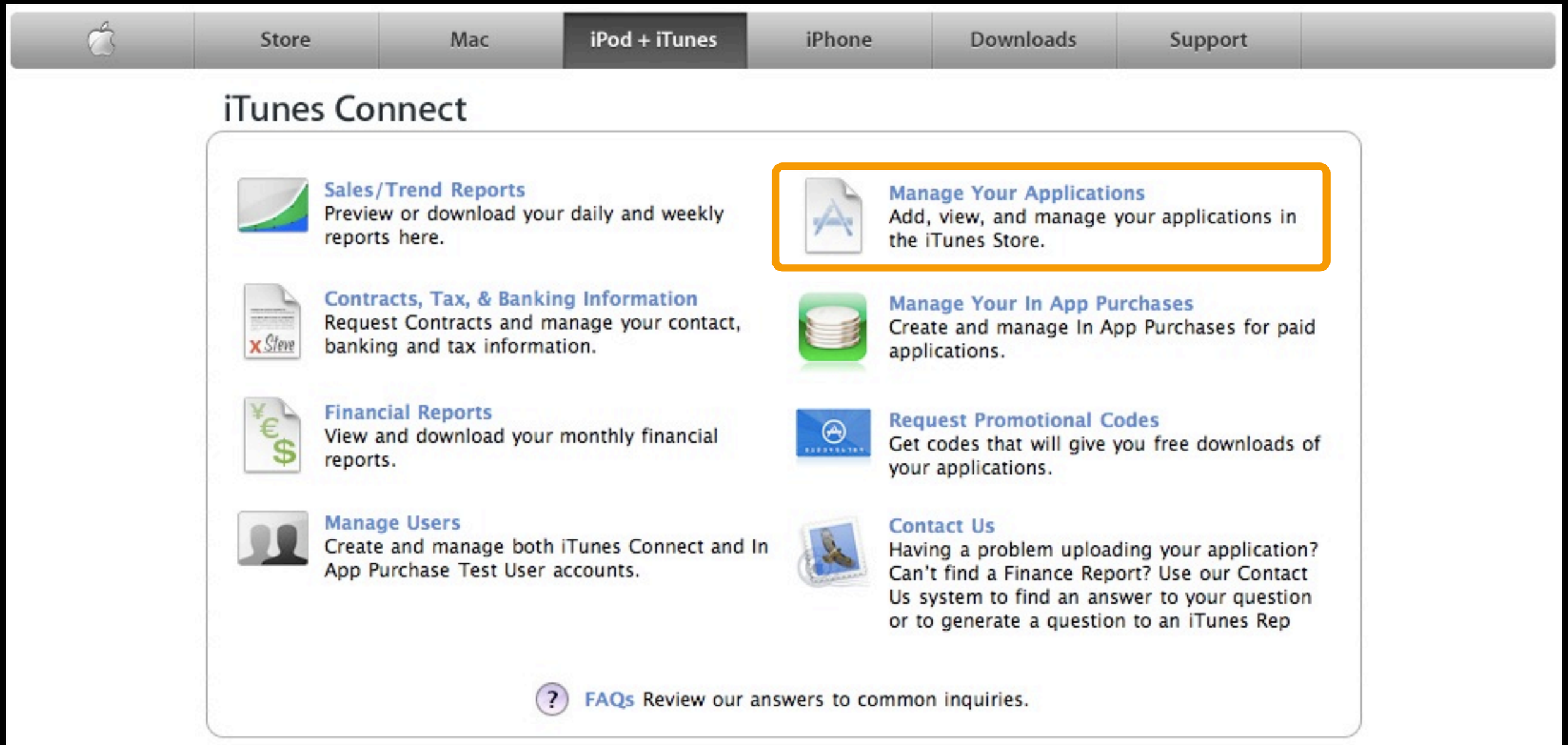
Implementing In App Purchase

- Define metadata and upload to iTunes Connect
- Build purchase workflow
- Test in sandbox environment

Create Item List

Name	Description	Price Tier	Product ID
Item A	Product A provides…	Tier 1	com.example.app.1
Item B	Product B provides…	Tier 1	com.example.app.2
Item C	Product C provides…	Tier 2	com.example.app.3
Bundle Item 1	Bundle 1 provides A+B	Tier 5	com.example.app.4
…	…	…	…

Loading Items

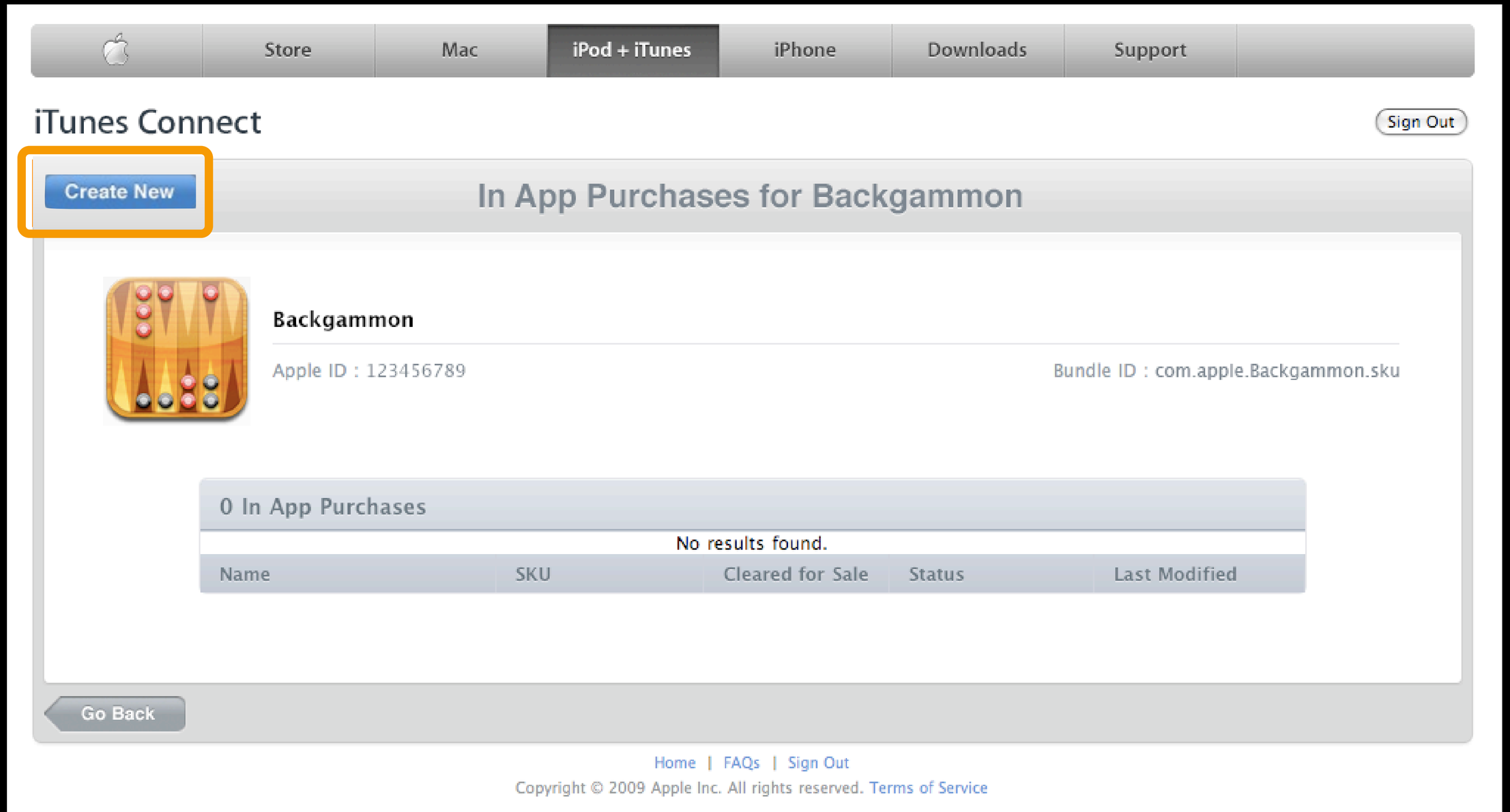


Managing Items

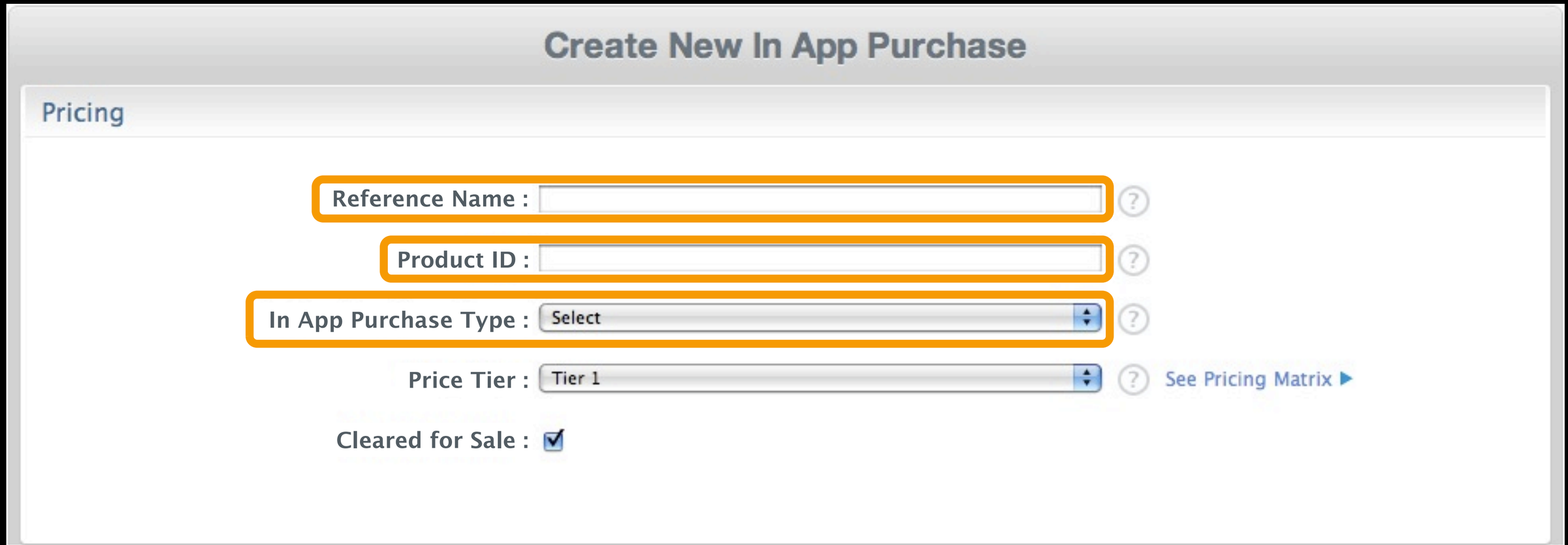
Backgammon
Version: 1.0
App Details
Date Submitted : 20 May 2008
Status: Ready for Sale

Edit Information | Reject Binary | Manage In App Purchases

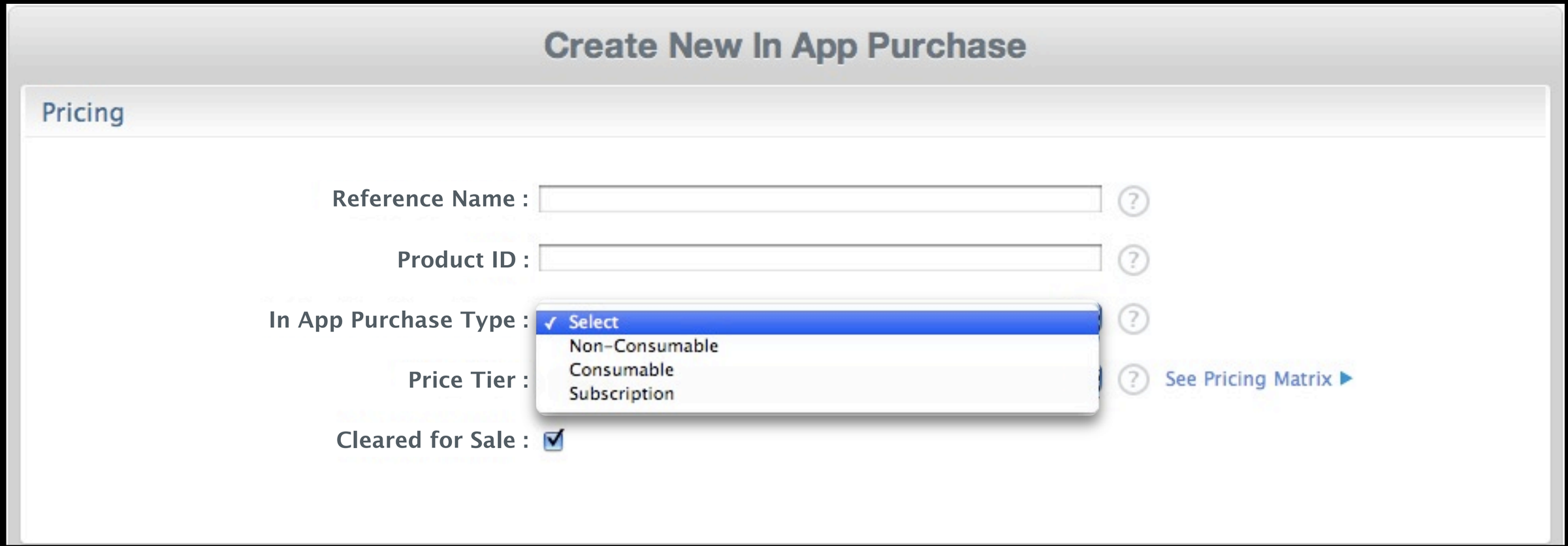
Create Item


Store
Mac
iPod + iTunes
iPhone
Downloads
Support
iTunes Connect
Sign Out
Create New
In App Purchases for Backgammon
Backgammon
Apple ID : 123456789
Bundle ID : com.apple.Backgammon.sku
0 In App Purchases
No results found.
Name
SKU
Cleared for Sale
Status
Last Modified
Go Back
Home | FAQs | Sign Out
Copyright © 2009 Apple Inc. All rights reserved. Terms of Service

Enter Item Metadata



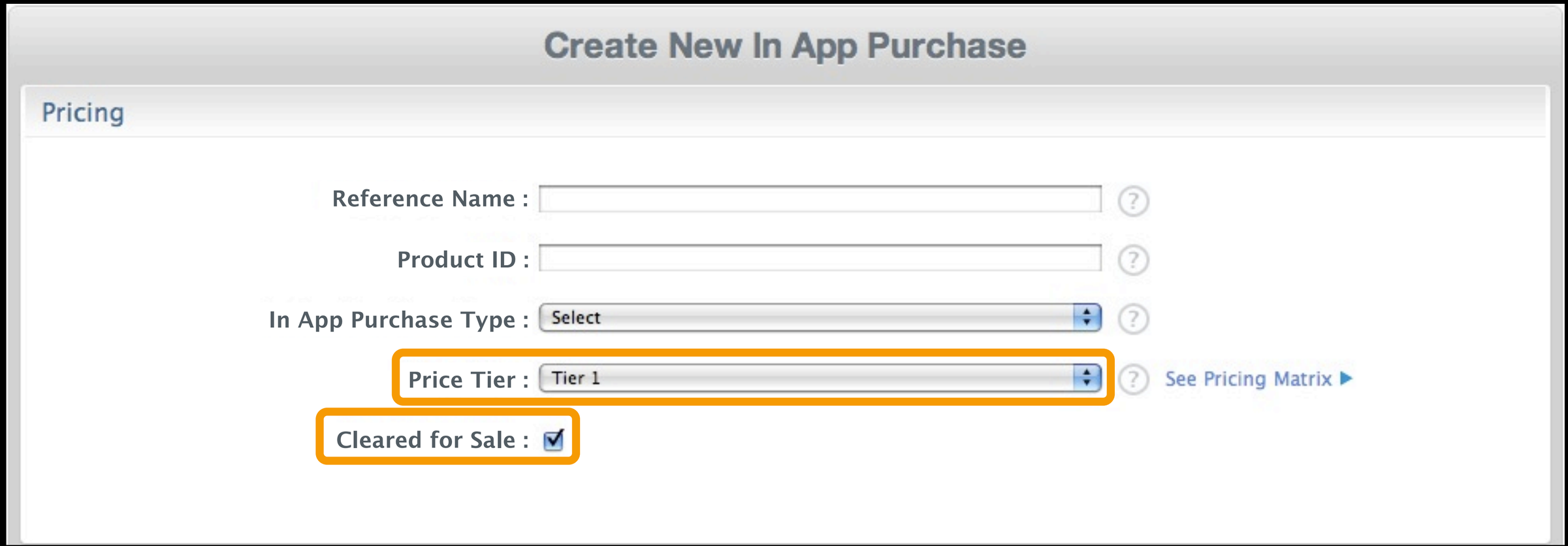
Enter Item Metadata



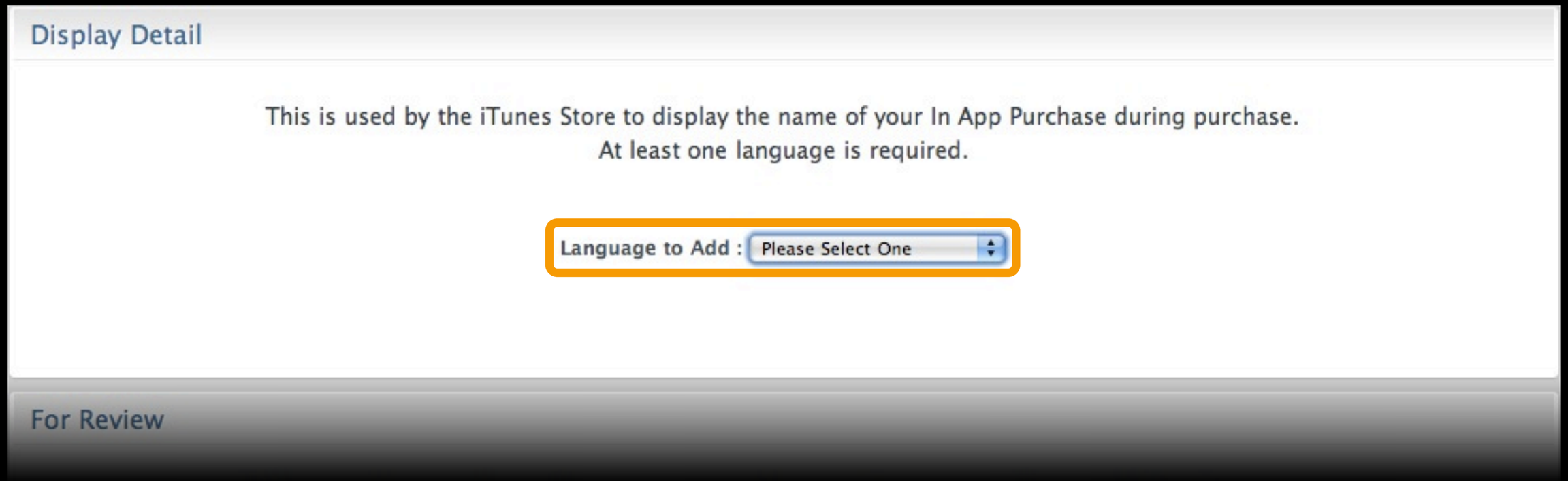
In App Purchase Types

- Nonconsumable—default
  - Persistent
  - Purchased once
  - Available across all devices
- Consumable
  - Used up or disappear after use
  - Can be purchased multiple times
  - Not required to be available across all devices
- Subscription
  - Same as consumable except—required to be shared across devices

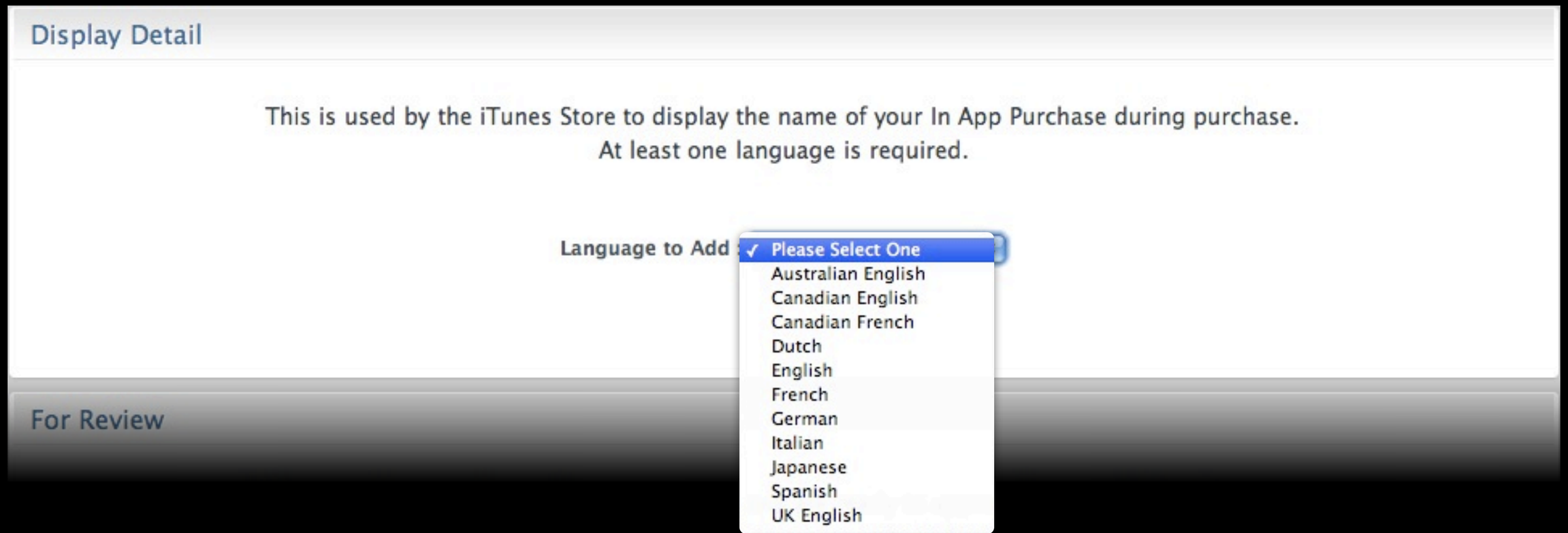
Enter Item Metadata



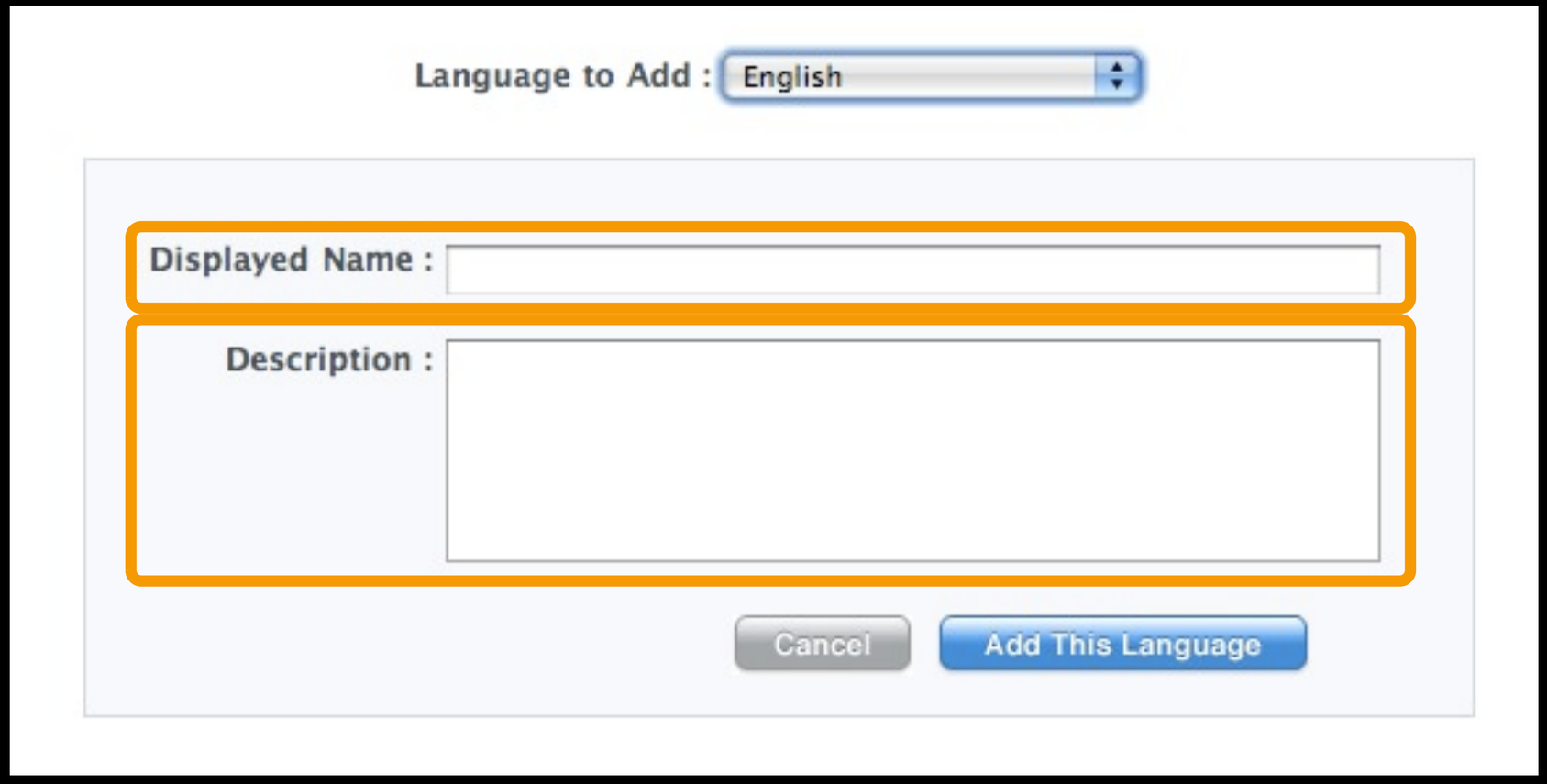
Choose Item Localizations



Choose Item Localizations



Edit Item Localization Details



Add Item Review Screenshot

For Review

When you have tested your In App Purchase and are ready to sign-off please upload a screenshot of your In App Purchase. This will be used for review purposes only.

Screenshot :

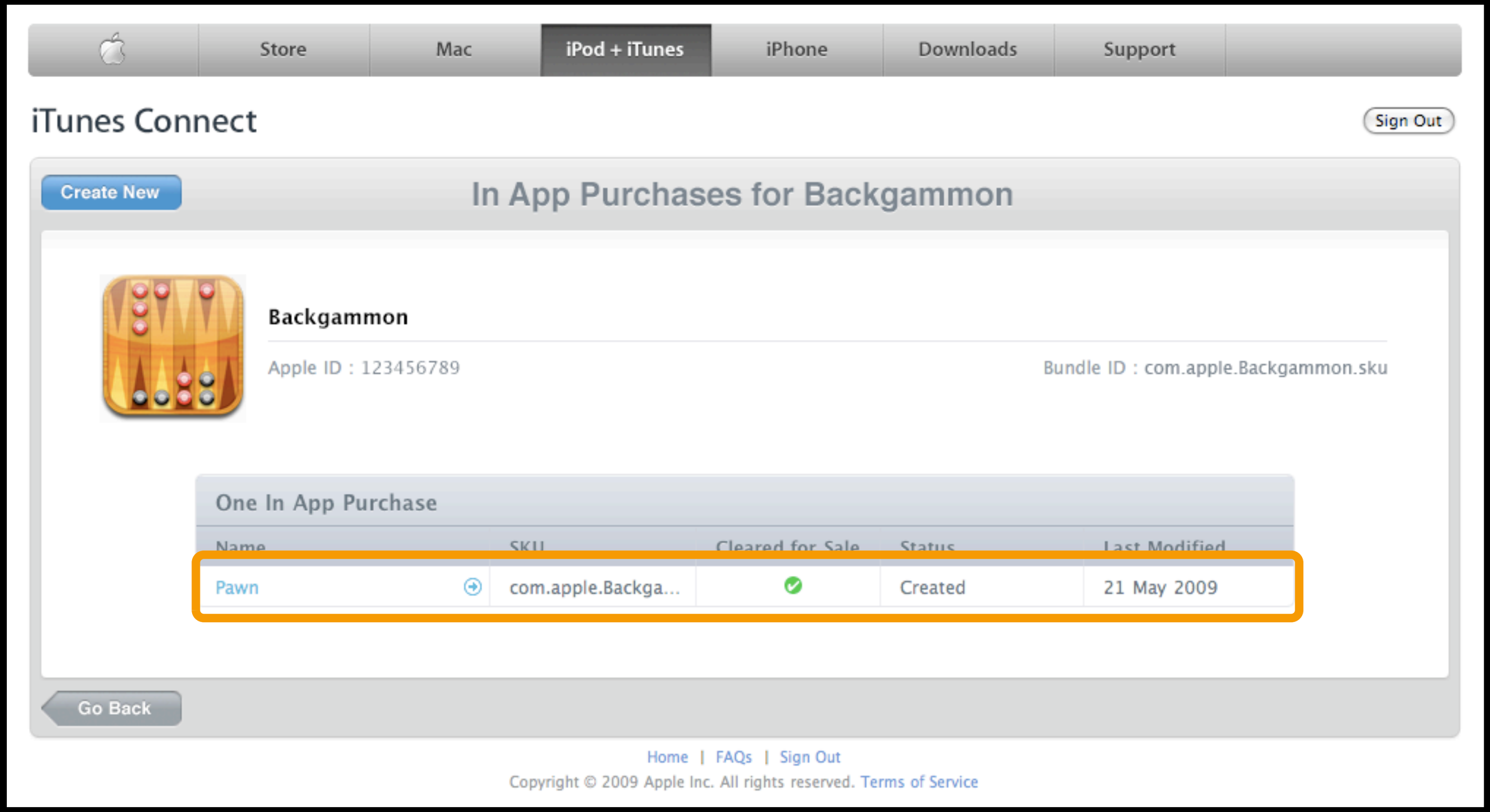
Screenshots must be 320x480, 480x320, 320x460 or 480X300 pixels.

Choose File

Cancel

Save Changes

Managing Items



Implementing In App Purchase

- Define metadata and upload to iTunes Connect
- Build purchase workflow
- Test in sandbox environment

Implementing In App Purchase

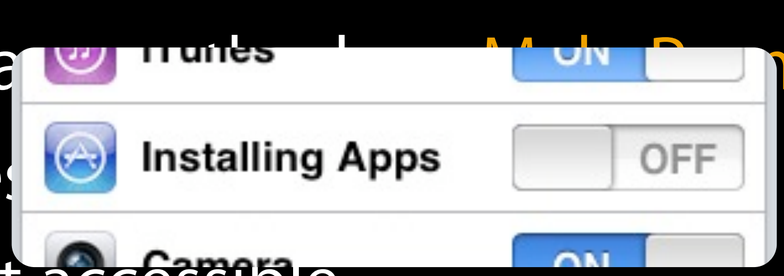
- Define metadata and upload to iTunes Connect
- Build purchase workflow
- Test in sandbox environment

Build Purchase Workflow

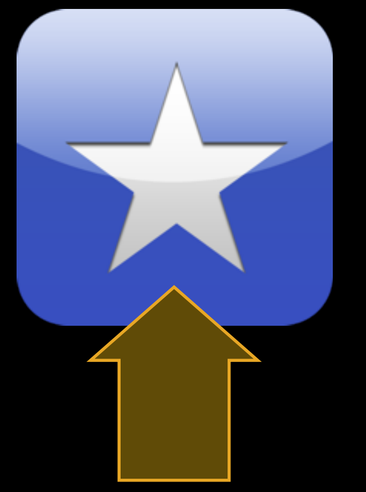
1. Validate App Store access
2. Load catalog
3. Fetch localized details
4. Present items for selection
5. Request item payment
6. Inspect payment transaction
7. Validate transaction
8. Unlock built-in items or provide remote access
9. Remember purchase
10. Support item restore

1 Validate App Store Access

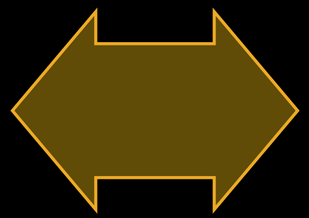
- Installing apps and purchased items may be prevented
  - Parental controls
  - Company Configuration Profile
- Call Payment Queue cla ments
  - YES—App Store acces
  - NO—App Store is not accessible



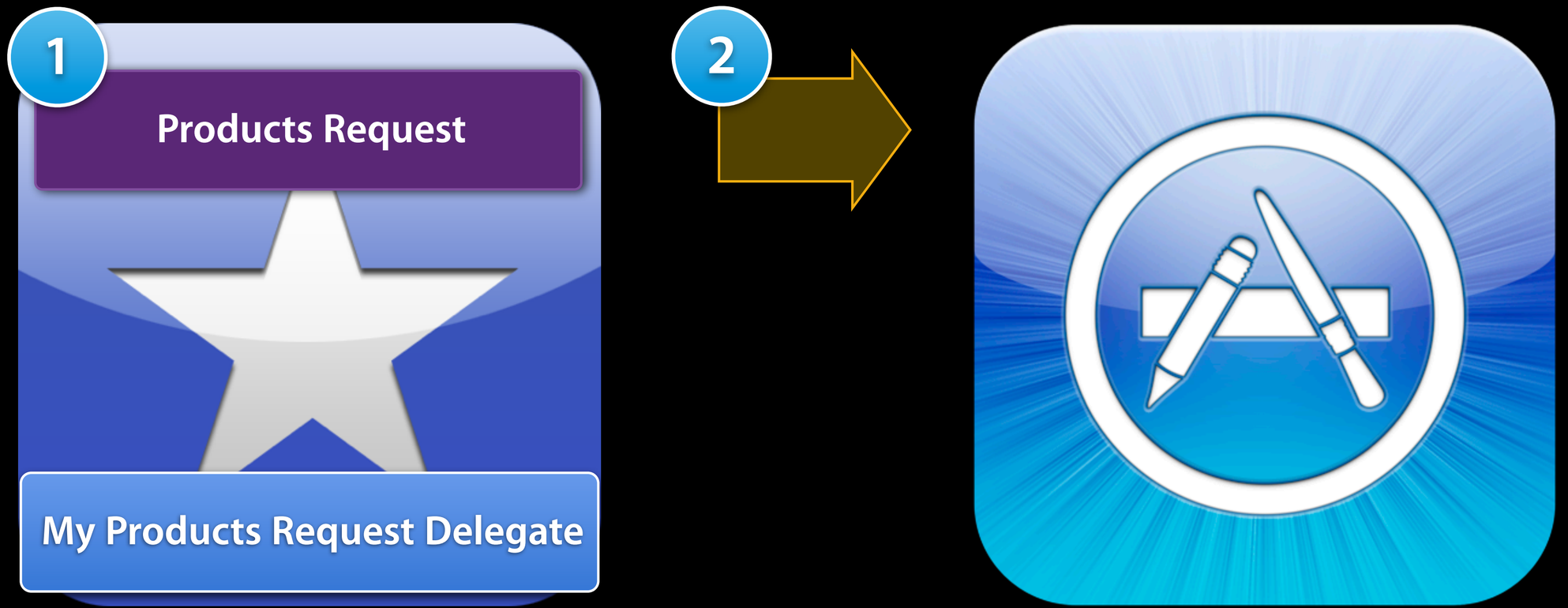
2 Load Built-in Catalog IDs



Load Remote Catalog IDs

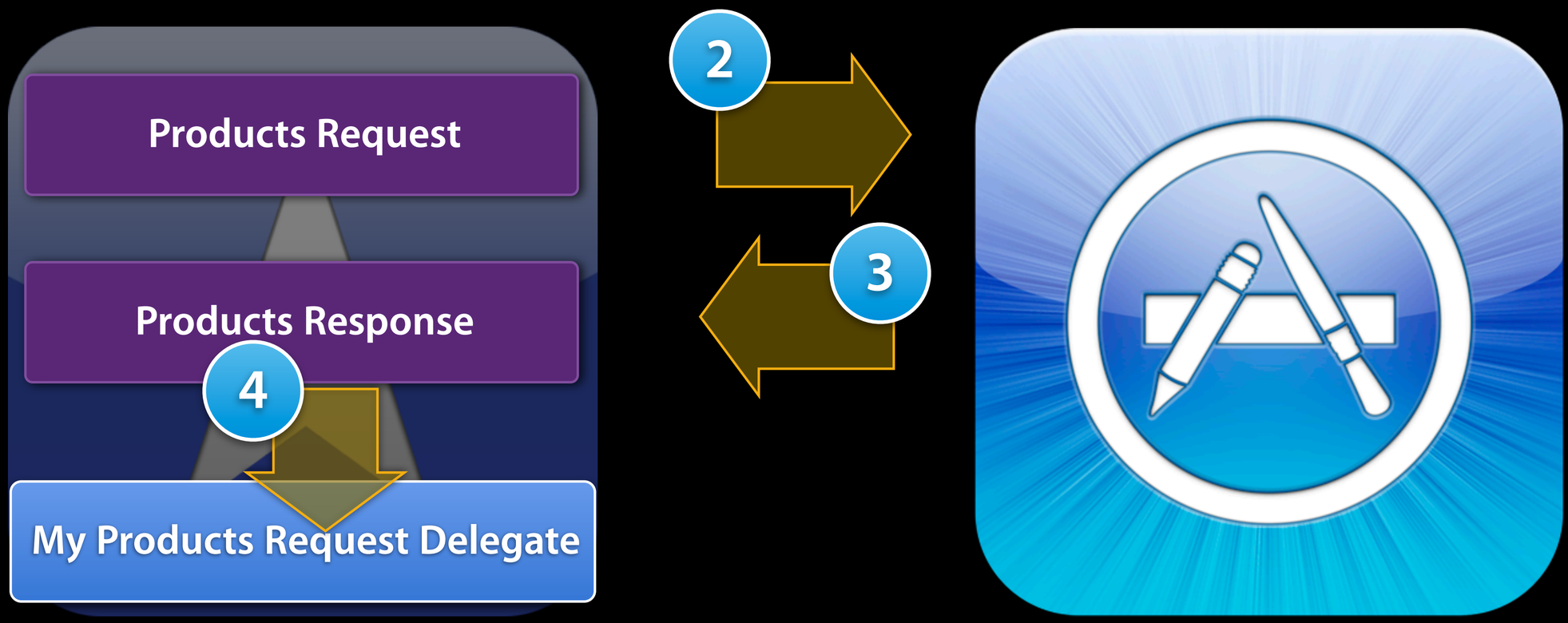


3 Fetch Localized Details




```
myPossibleProds = [NSSet setWithObjects: @”myProd1”, @”myProd2”, @”myProd3, nil];

myProdRequest = [[SKProductsRequest alloc] initWithProductIdentifiers: myPossibleProds];

myProdRequest.delegate = myProductsRequestDelegate;

[myProdRequest start];
```

3 Fetch Localized Details




```
-(void)productsRequest:(SKProductsRequest *)request didReceiveResponse:(SKProductsResponse *)response
```

Fetch Localized Details

```
response.products

response.invalidProductIdentifiers
```

- Invalid products
  - Never entered into iTunes Connect
  - Aren't marked as "Cleared for Sale"
  - Haven't propagated to all servers

4 Build Item Display

response.products

response.invalidProductIdentifiers

```
myAvailableProductsArray = response.products;
for(SKProduct* aProduct in myAvailableProductsArray) {
  diplayTitle  = aProduct.localizedTitle;
  displayDesc  = aProduct.localizedDescription;
  displayPrice = aProduct.price;
}
```

5 Present Items for Selection

Present Items for Selection

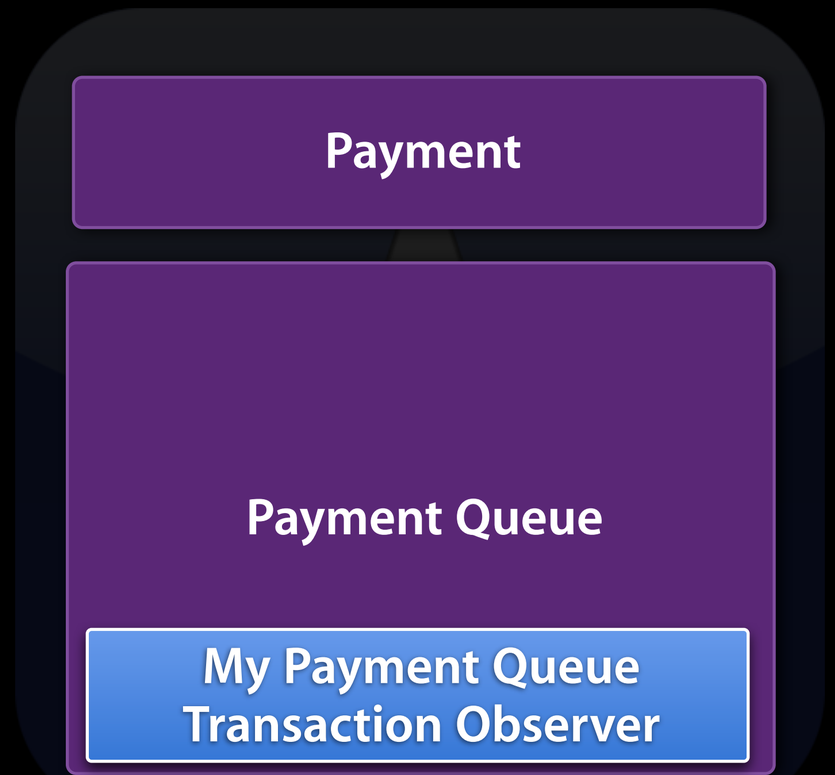
6 Request Payment

```
myPayment = [SKPayment paymentForProduct: selectedProduct userInfo: nil];
```

or

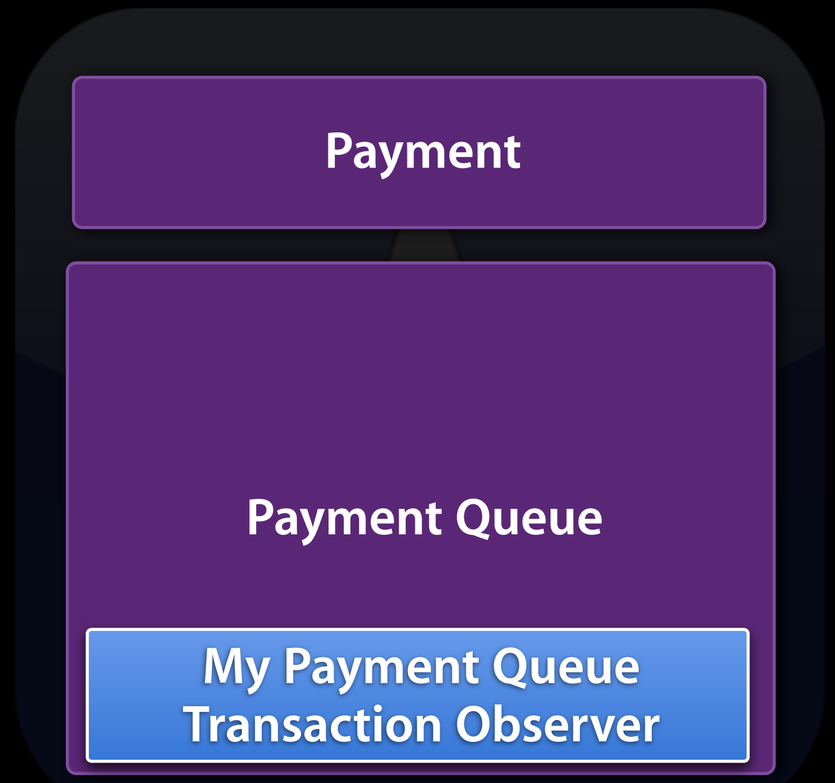
```
myPayment = [SKPayment paymentForProductIdentifier: selectedProdID userInfo: nil];
```

Request Item Payment



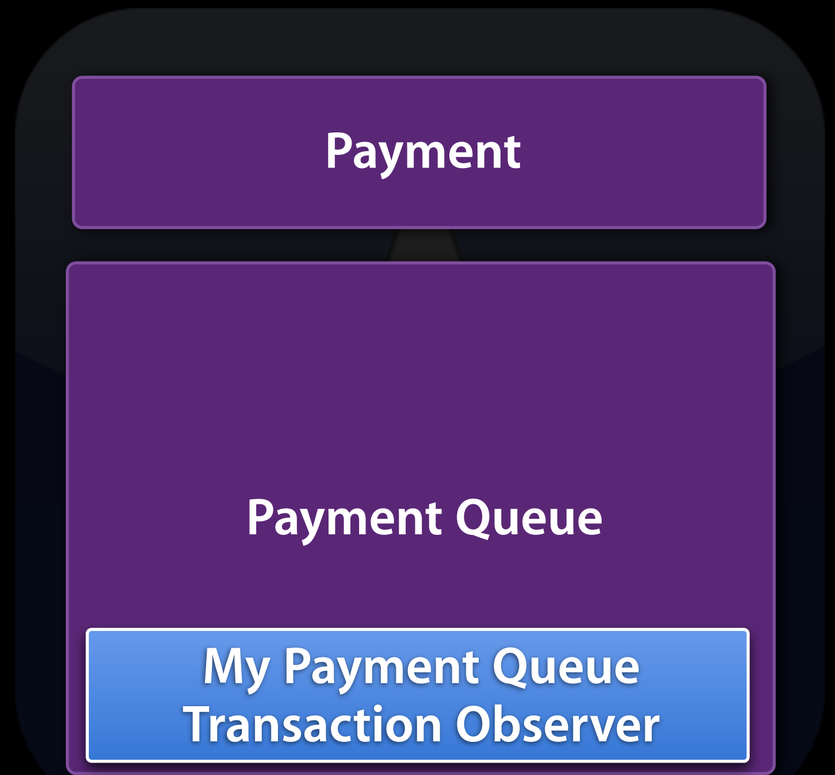

```
defaultPaymentQueue = [SKPaymentQueue defaultQueue];
[defaultPaymentQueue addTransactionObserver: myObserver];
```

Request Item Payment



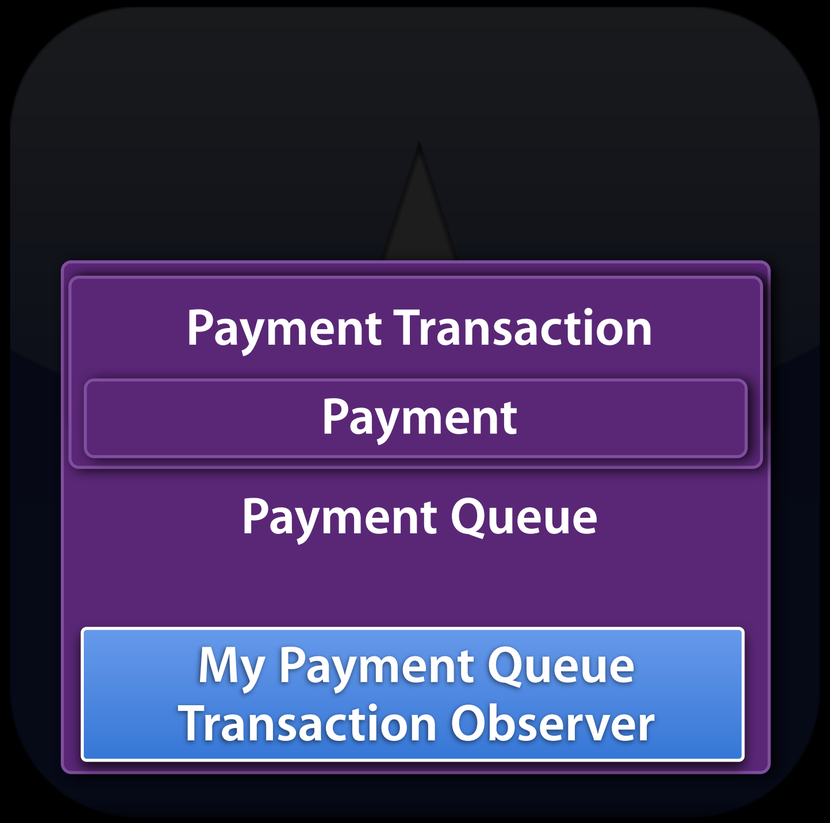

```
defaultPaymentQueue = [SKPaymentQueue defaultQueue];
[defaultPaymentQueue addTransactionObserver: myObserver];
[defaultPaymentQueue addPayment: myPayment];
```

Request Item Payment




```
defaultPaymentQueue = [SKPaymentQueue defaultQueue];
[defaultPaymentQueue addTransactionObserver: myObserver];
[defaultPaymentQueue addPayment: myRequest];
```

Request Item Payment




```
defaultPaymentQueue = [SKPaymentQueue defaultQueue];
[defaultPaymentQueue addTransactionObserver: myObserver];
[defaultPaymentQueue addPayment: myRequest];
```

Request Item Payment

Payment Transaction

Payment

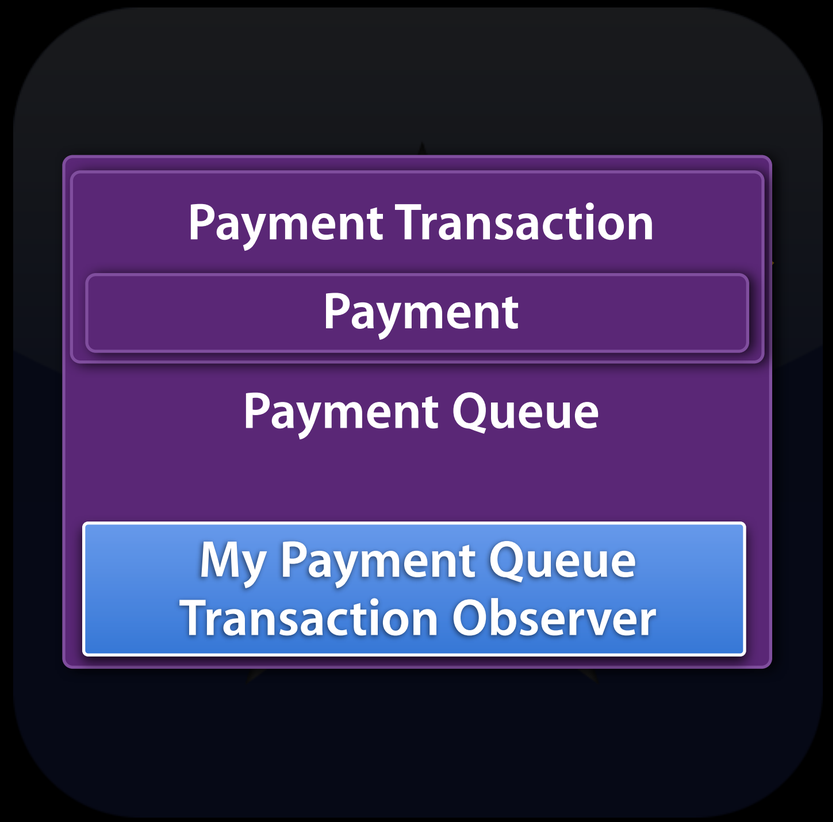
Payment Queue

My Payment Queue
Transaction Observer

Purchase?

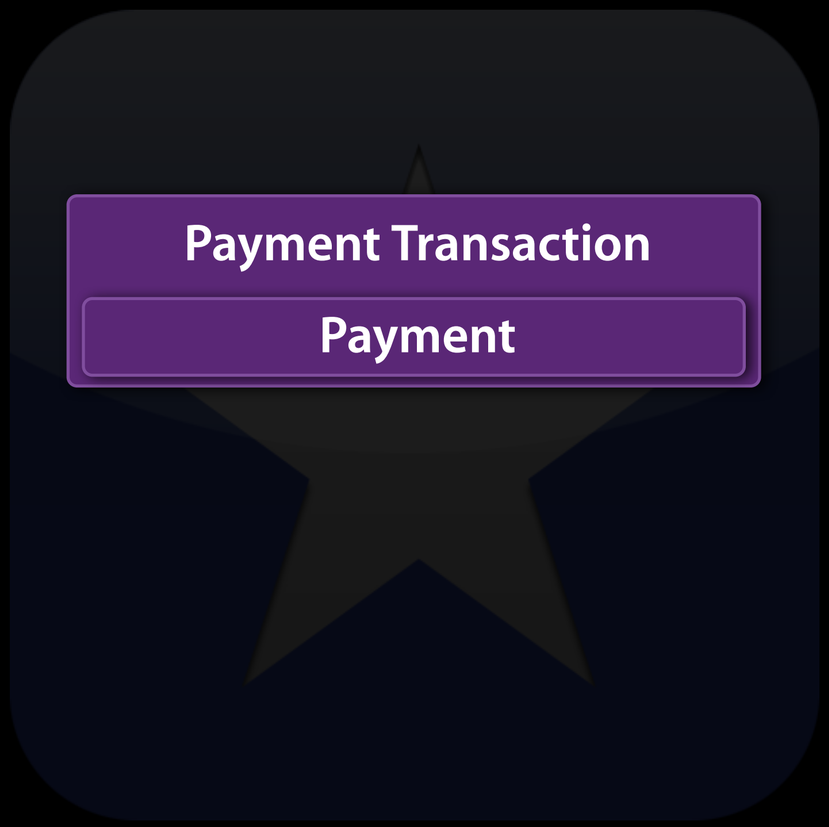
1 Product B @ $1.99

Request Item Payment



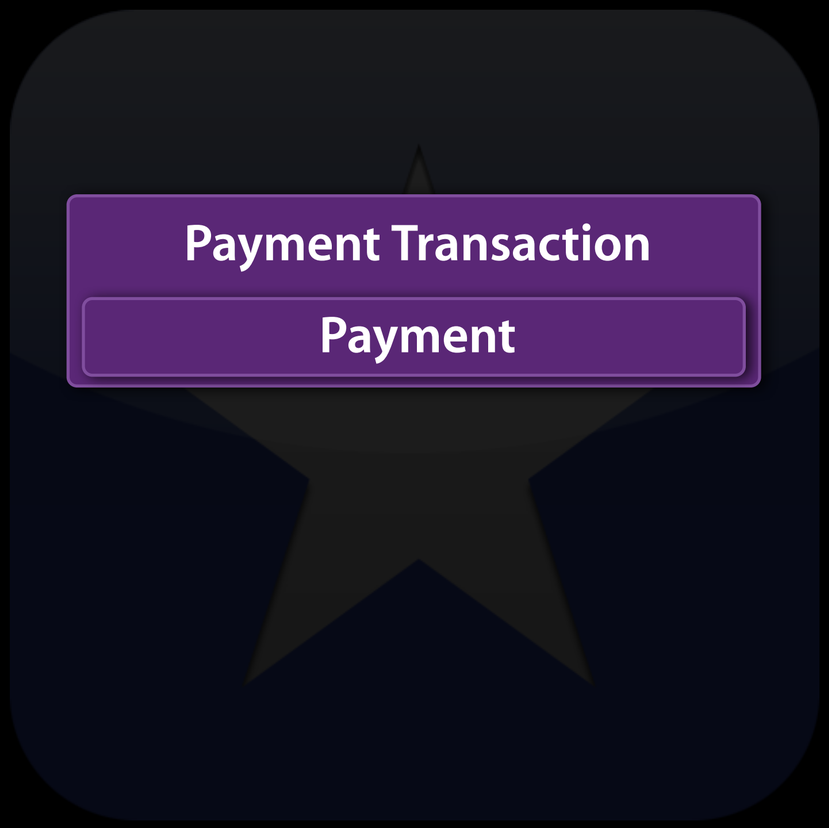

```
- (void)paymentQueue:(SKPaymentQueue *)queue updatedTransactions:(NSArray *)transactions
```

7 Validate Payment Transaction



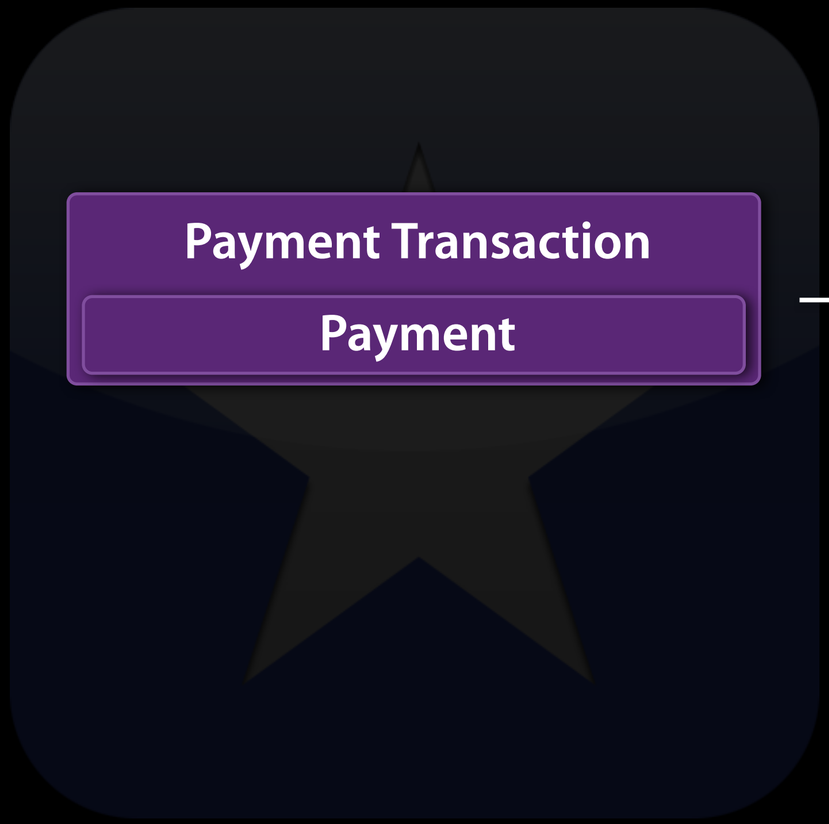

```
for(SKPaymentTransaction *aTransaction in transactions) {
  switch (aTransaction.transactionState ) {

    // item is in the process of being purchased
    case SKPaymentTransactionStatePurchasing:
                ...
    break;

    // item was successfully purchased
    case SKPaymentTransactionStatePurchased:
                ...
    break;

    // already purchased this product
    case SKPaymentTransactionStateRestored:
                ...
    break;

    // user canceled the transaction
    case: SKPaymentTransactionStateFailed
                ...
    break;
...
```

Validate Payment Transaction




```
for(SKPaymentTransaction *aTransaction in transactions) {
  switch (aTransaction.transactionState ) {

    // item is in the process of being purchased
    case SKPaymentTransactionStatePurchasing:
                ...
    break;

    // item was successfully purchased
    case SKPaymentTransactionStatePurchased:
                ...
    break;

    // already purchased this product
    case SKPaymentTransactionStateRestored:
                ...
    break;

    // user canceled the transaction
    case: SKPaymentTransactionStateFailed
                ...
    break;
...
```

Validate Payment Transaction




```
aTransaction.transactionReceipt
```

- Used to validate e-commerce transaction
- Cryptographically signed chunk of data
- Contains information about the purchase
- Can validate using Web-based API

Validate Payment Transaction

Your Server

Apple Server

Request Payment

Transaction Receipt

Post Transaction Receipt

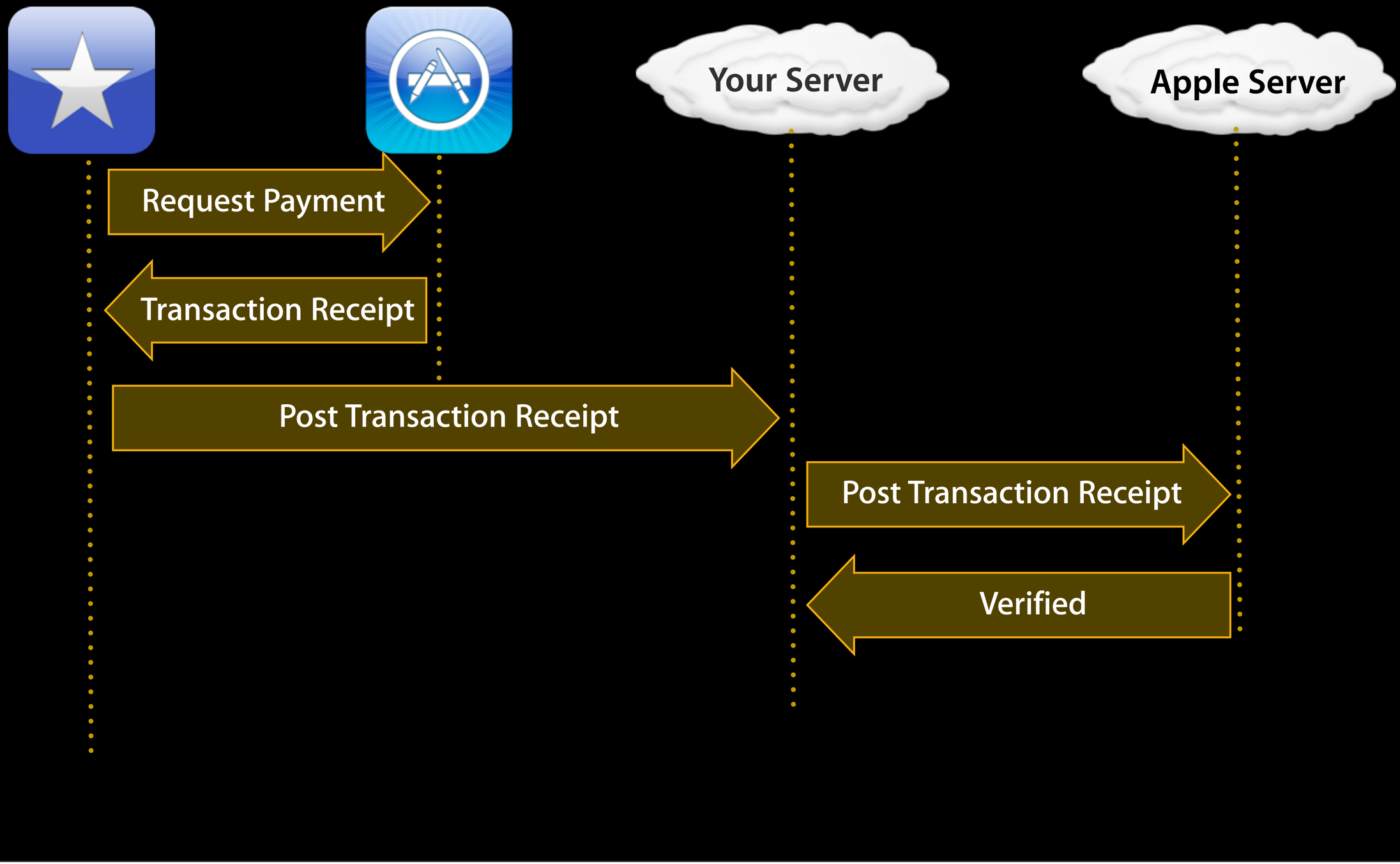
Apple Validation URL

https://buy.itunes.apple.com/verifyReceipt

Validation Post Data

```
{ receipt-data : ...}
```

Validate Payment Transaction



Validation Reponse

```
{ status : 0 }
```

```
{ status : any_other_value }
```

Validate Payment Transaction

Your Server

Apple Server

Request Payment

Transaction Receipt

Post Transaction Receipt

Post Transaction Receipt

Verified

Verified

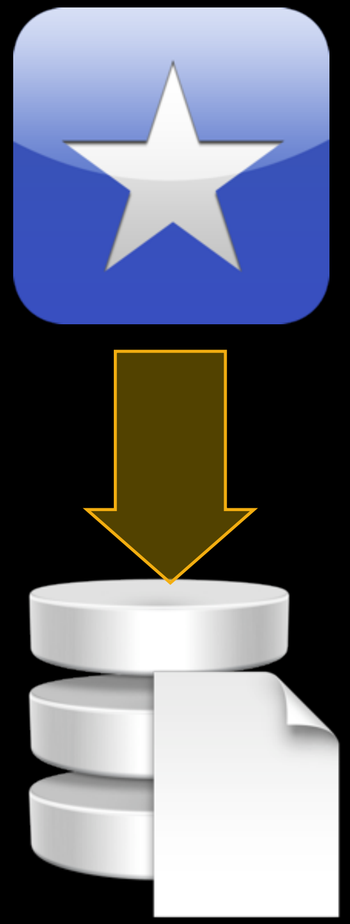
Unlock Item

8 Unlock Purchased Items

- Customers may quit your app anytime
- Store Kit Payment Queue is persistent
- Transactions are persistent
- Confirm access
- Removing transaction

```
[[SKPaymentQueue defaultQueue] finishTransaction: transaction];
```

9 Remember Purchase



10 Support Item Restore

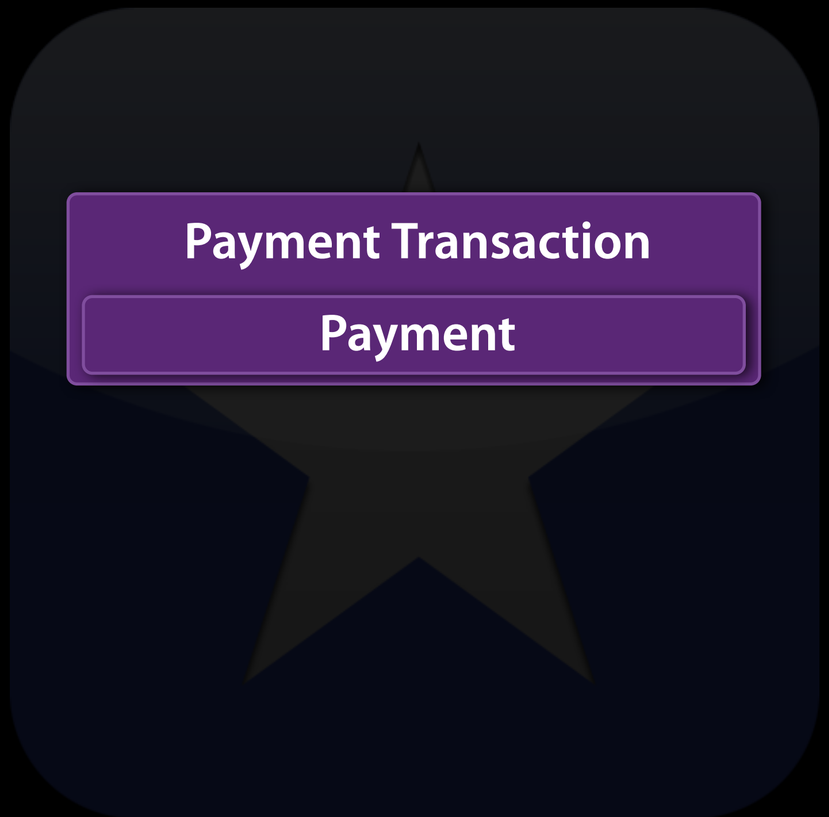
- Like apps, purchased items are only paid for once
- Available across customer devices
- Available after app is deleted then reinstalled

Apps Available Across Devices

Items Available Across Devices

Restored After Reinstall

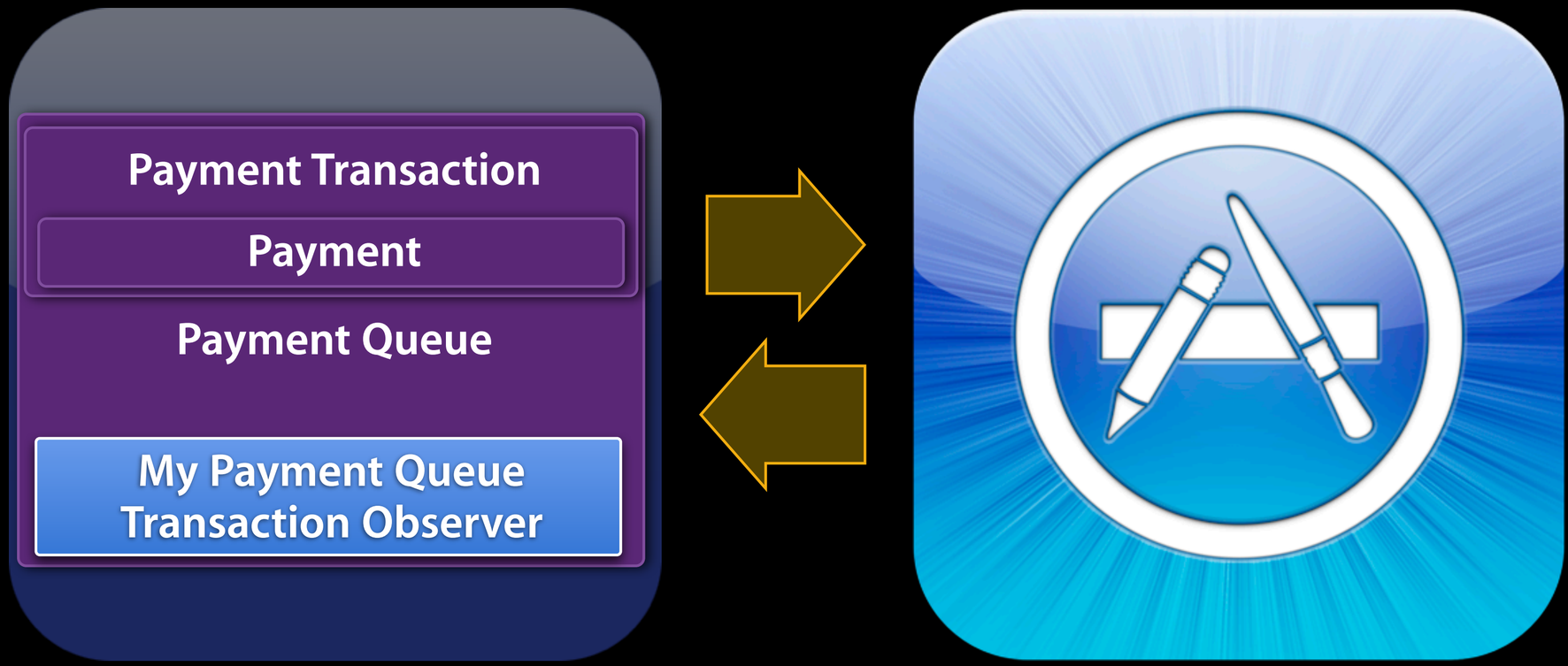
Support Item Restore




```
for(SKPaymentTransaction *aTransaction in transactions) {
  switch (aTransaction.transactionState ) {

    // item is in the process of being purchased
    case SKPaymentTransactionStatePurchasing:
            ...
    break;

    // item was successfully purchased
    case SKPaymentTransactionStatePurchased:
            ...
    break;

    // already purchased this product
    case SKPaymentTransactionStateRestored:
            ...
    break;

    // user canceled the transaction
    case: SKPaymentTransactionStateFailed
            ...
    break;
...
```

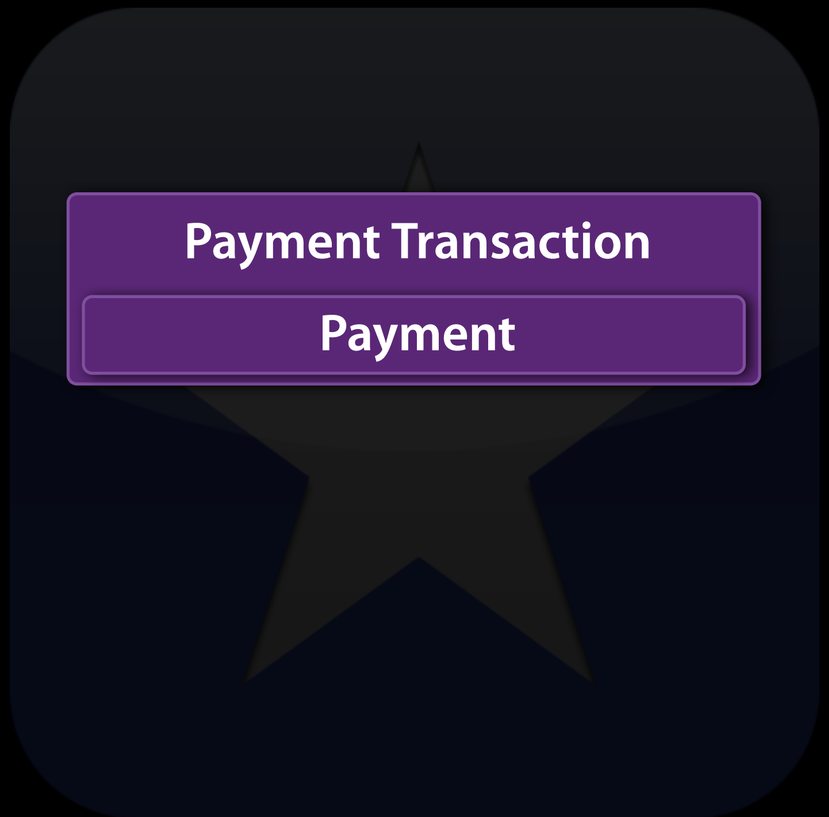
Support Item Restore

Support Item Restore




```
defaultPaymentQueue = [SKPaymentQueue defaultQueue];
[defaultPaymentQueue restoreCompletedTransactions];
```

Support Item Restore




```
for(SKPaymentTransaction *aTransaction in transactions) {
  switch (aTransaction.transactionState ) {

    // item is in the process of being purchased
    case SKPaymentTransactionStatePurchasing:
            ...
    break;

    // item was successfully purchased
    case SKPaymentTransactionStatePurchased:
            ...
    break;

    // already purchased this product
    case SKPaymentTransactionStateRestored:
            ...
    break;

    // user canceled the transaction
    case: SKPaymentTransactionStateFailed
            ...
    break;
...
```

Build Purchase Workflow

1. Validate App Store access
2. Load catalog
3. Fetch localized details
4. Present items for selection
5. Request item payment
6. Inspect payment transaction
7. Validate transaction
8. Unlock built-in items or provide remote access
9. Remember purchase
10. Support item restore

Implementing In App Purchase

- Define metadata and upload to iTunes Connect
- Build purchase workflow
- Test in sandbox environment

Implementing In App Purchase

- Define metadata and upload to iTunes Connect
- Build purchase workflow
- Test in sandbox environment

In App Sandbox

- For testing In App Purchase only
  - Accessible from device only—not simulator
  - No invoices sent

In App Sandbox

- For testing In App Purchase only

In App Sandbox

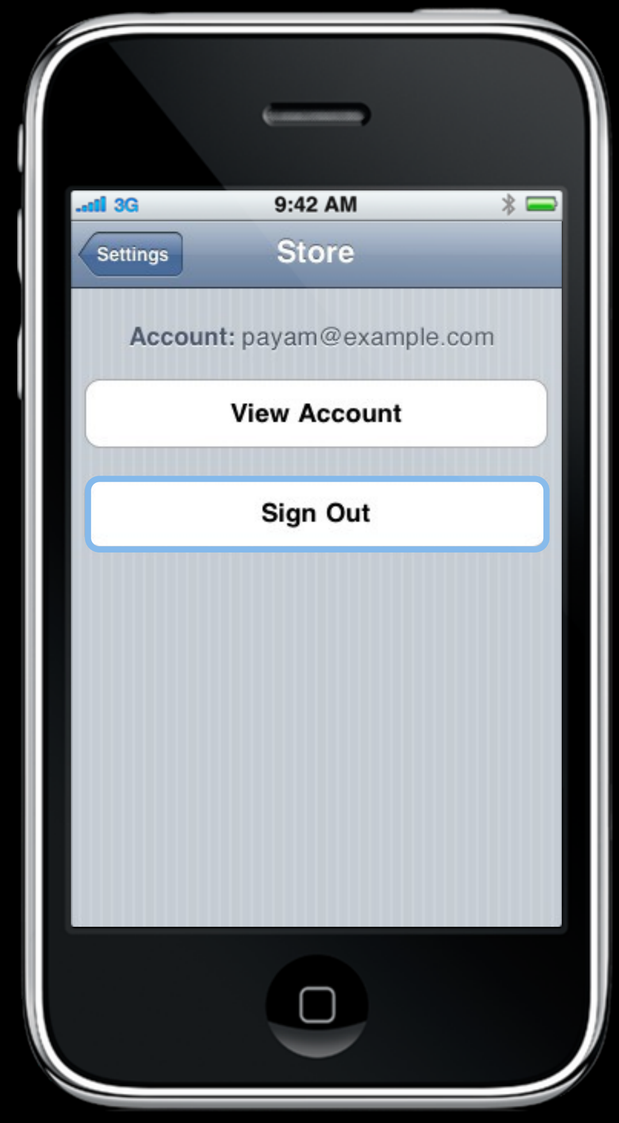
- For testing In App Purchase only
- Separate Apple ID required
  - Tied to a specific country, e.g. USA, UK
  - Cannot be used to test any other iTunes Store feature
  - Requires unique email address, consider using +
    - payam+usa@example.com—for testing on US items
    - payam+uk@example.com—for testing on Britian store items
    - payam+jp@example.com—for testing on Japan store items

In App Sandbox

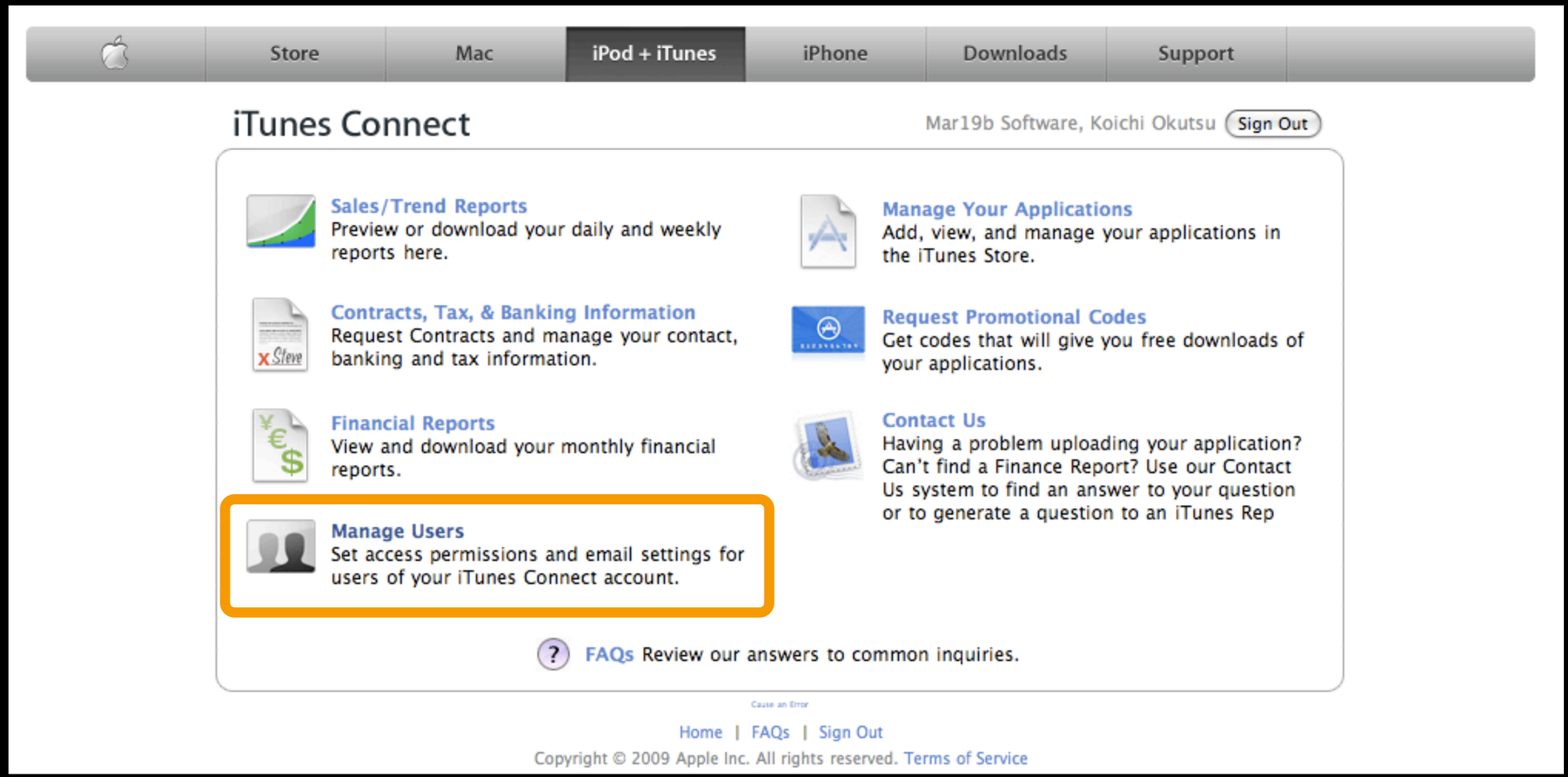
- For testing In App Purchase only
- Separate Apple ID required

In App Sandbox

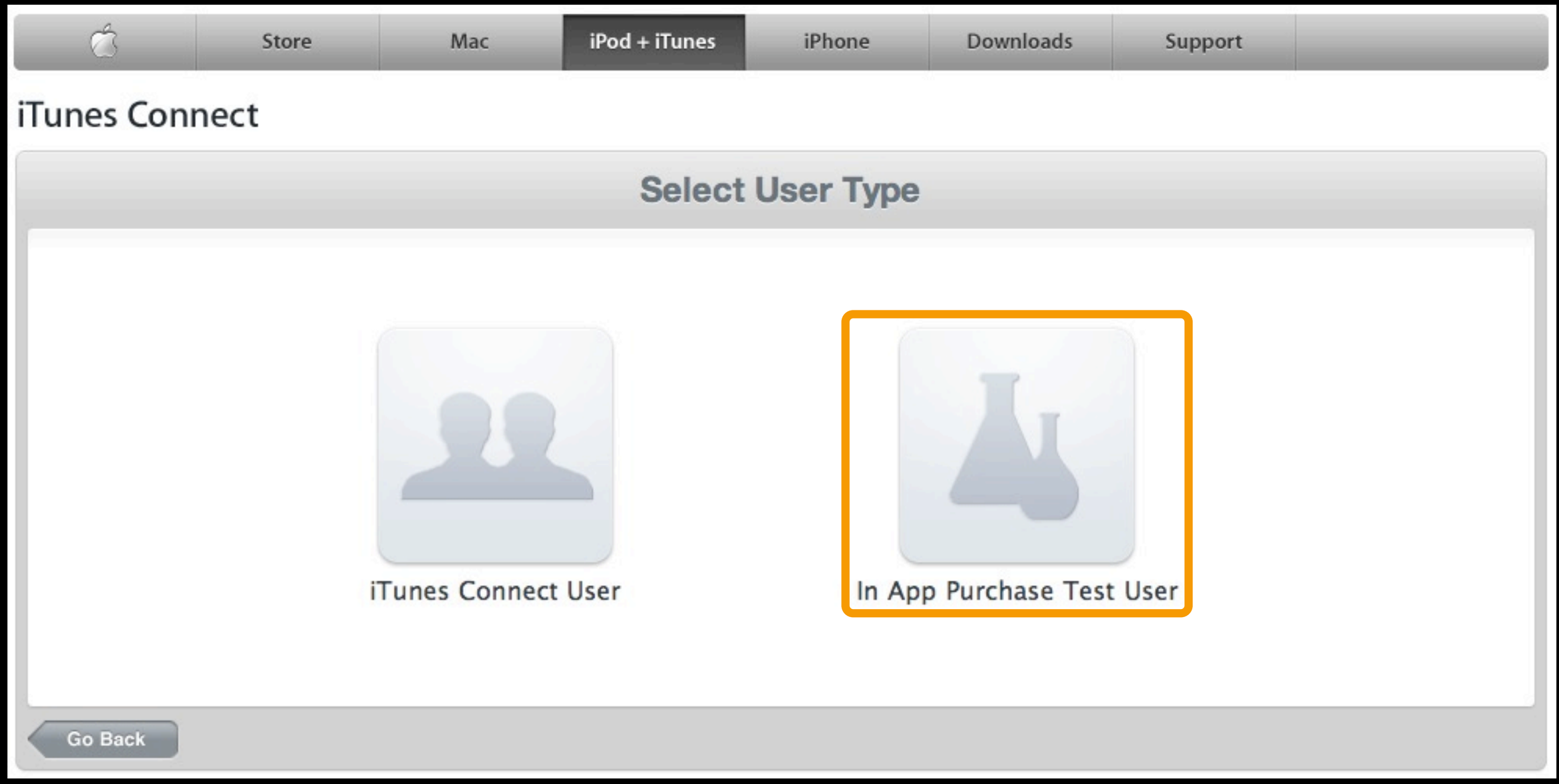
- For testing In App Purchase only
- Separate Apple ID required
- Verify store settings before testing
  - Sign out
  - Then launch your app to test



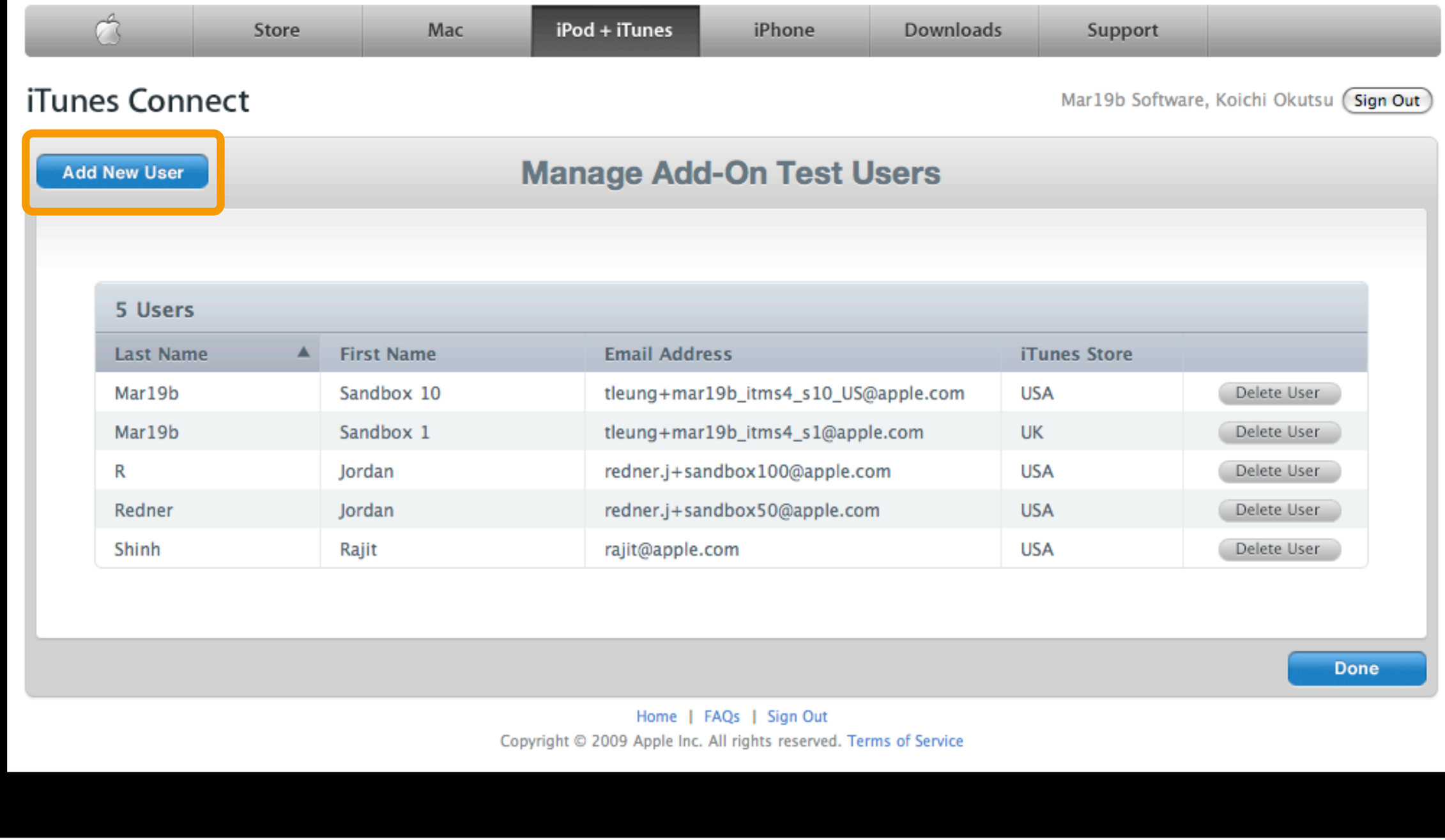
Manage Users



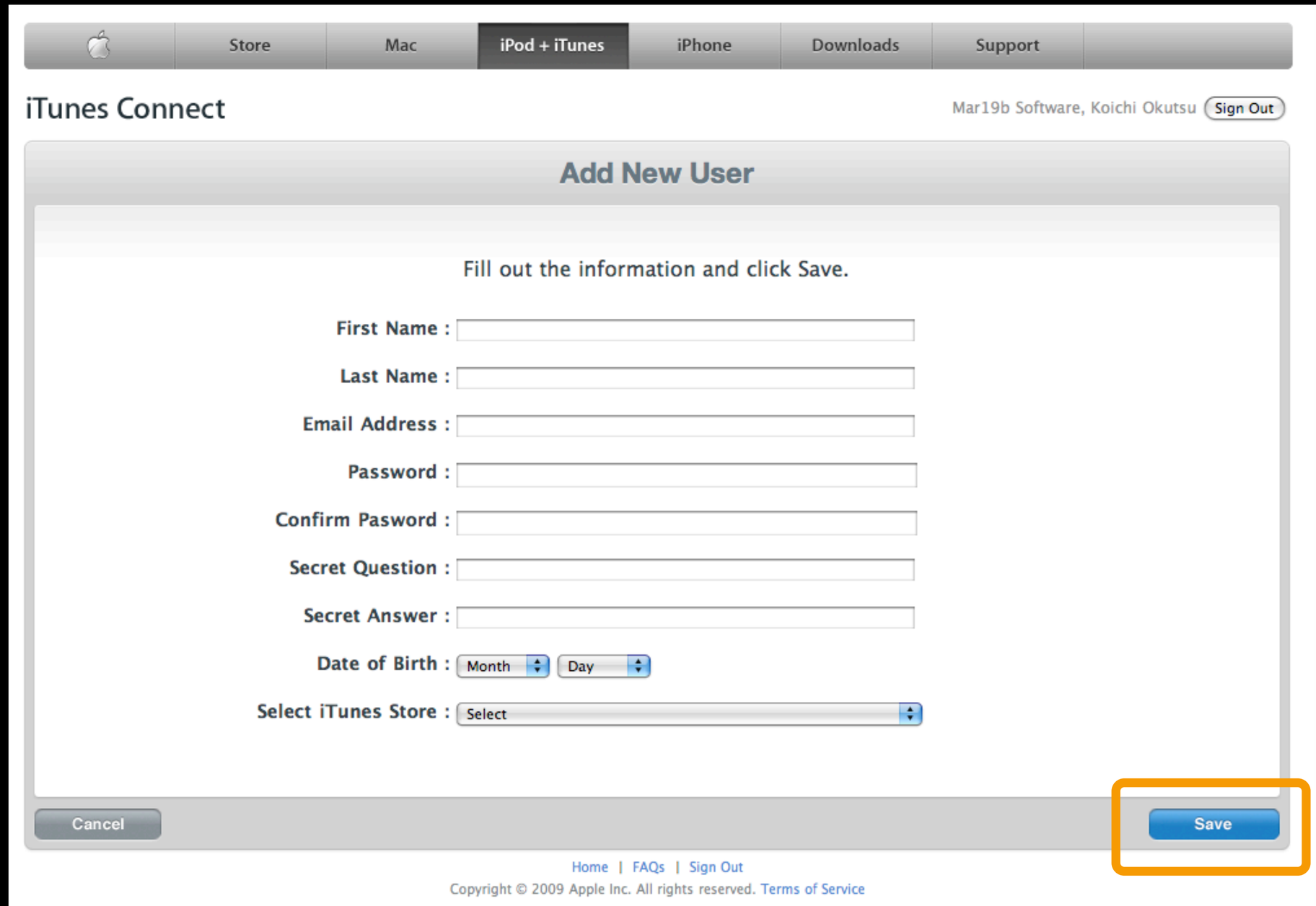
Select In App Purchase User



Add Sandbox User



Provide User Details



Implementing In App Purchase

- Define metadata and upload to iTunes Connect
- Build purchase workflow
- Test in sandbox environment

Implementing In App Purchase

- Define metadata and upload to iTunes Connect
- Build purchase workflow
- Test in sandbox environment

Financials

Financials

- Rules same as application financials
  - You get 70%
  - Payment schedule, royalties, reports are all the same
- Reports
  - Same report format
  - Vendor Identifier field contains item product ID
  - Parent Identifier field contains the ID of your app

Agenda

- Overview of In App Purchase
- Implementing In App Purchase
- Financial reports and more

Related Session and Lab

Publishing on the App Store	Presidio Friday 2:00 PM
iPhone In-App Purchase Lab	iPhone Lab C Wednesday 2:00 PM

More Information

Mark Malone
Internet and Server Technologies Evangelist
mmalone@apple.com

Documentation
iPhone Dev Center
http://developer.apple.com/iphone

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09